

Server-Side Web Programming: Python (Part 1)

Copyright © 2026 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - Python WSGI programming
 - Sending arguments to WSGI programs

Agenda

- **WSGI**
- Using WSGI
- URL encoding
- HTTP query strings

WSGI

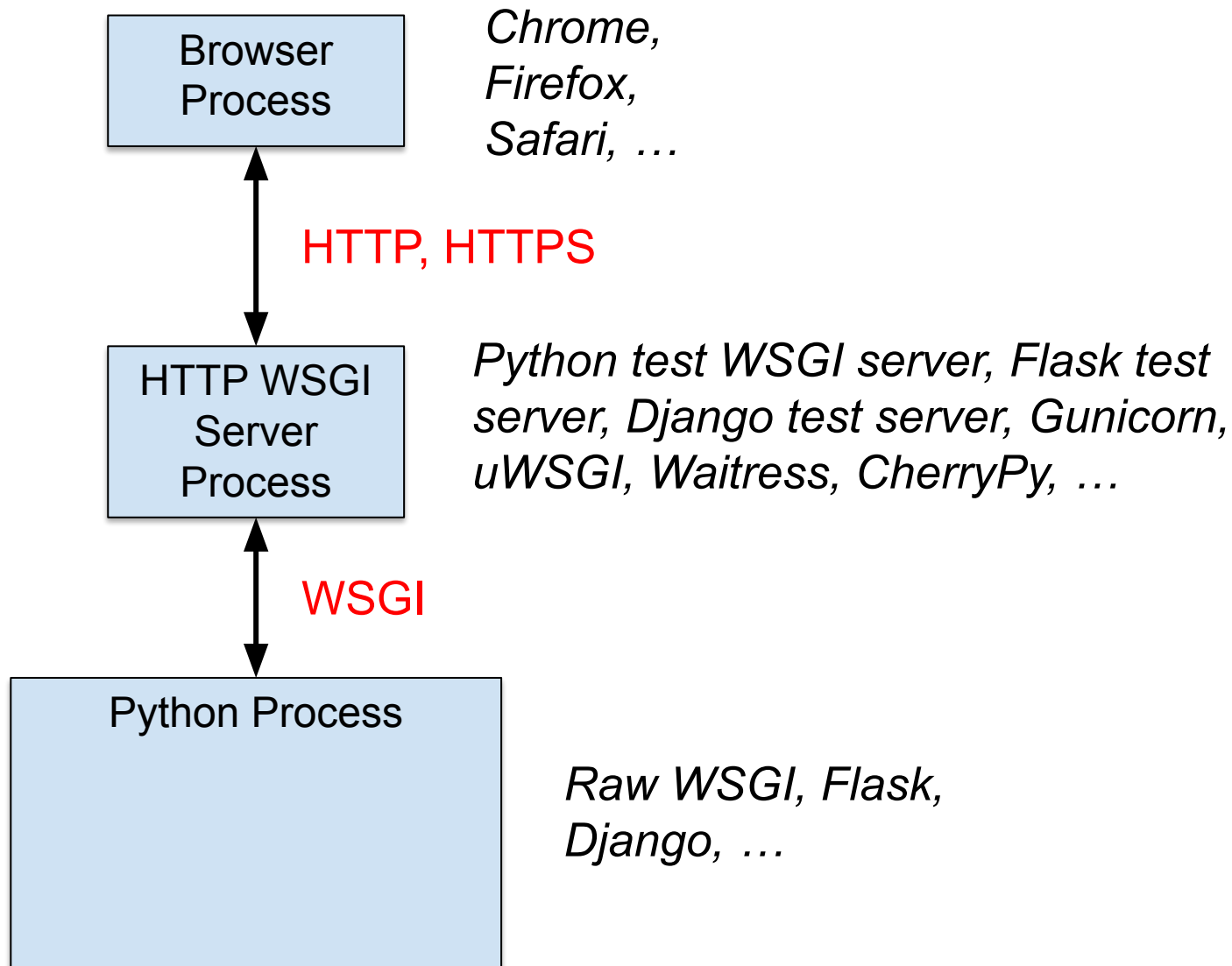
- **Problem:**

- Often HTTP server must generate data *dynamically*
 - Instead delivering a file containing data...
 - It must run a program that creates data, which it then delivers
 - Sometimes by accessing a DB

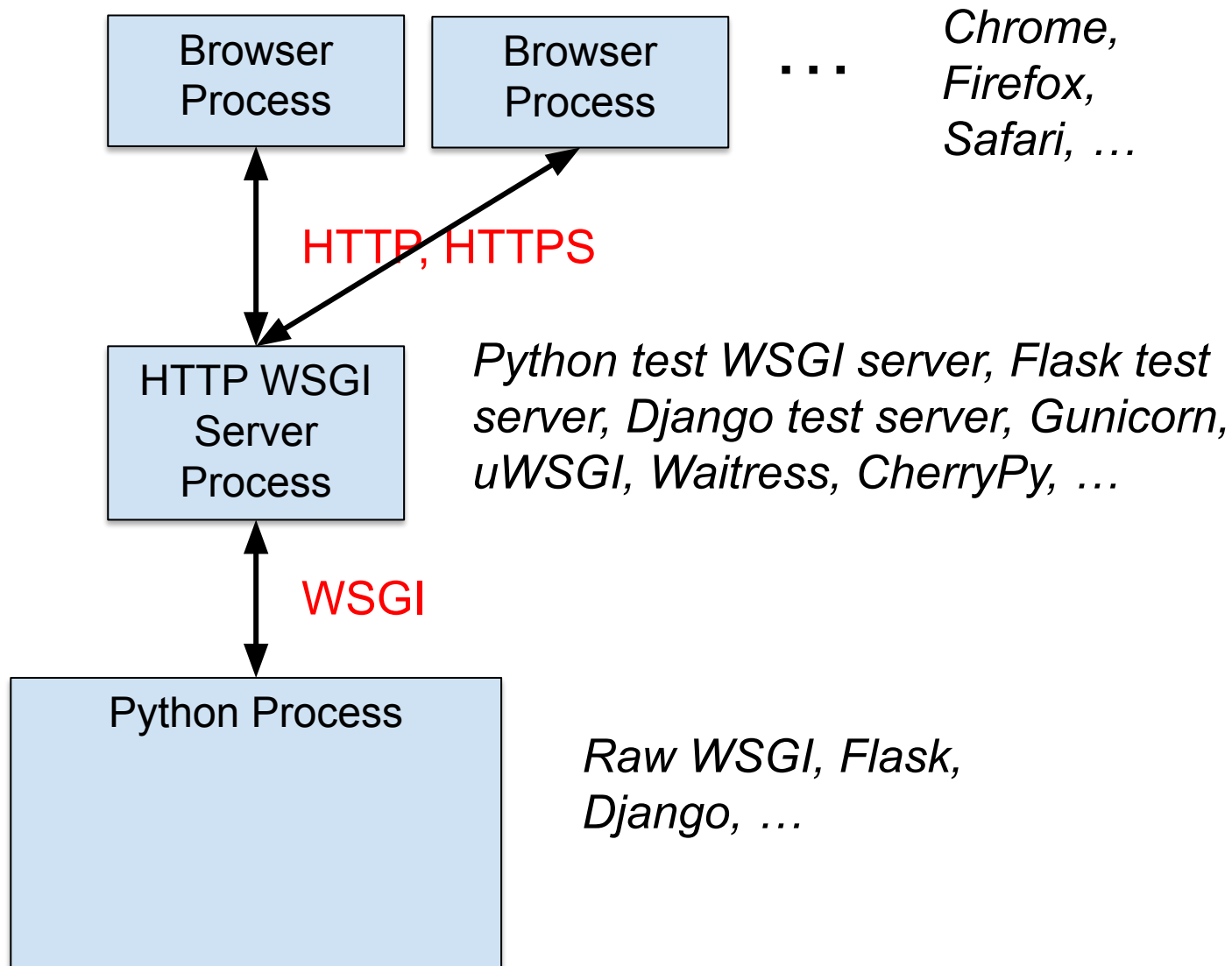
- **Python solution:**

- ***Web Server Gateway Interface (WSGI)***
 - A standard interface for HTTP servers to communicate with Python programs

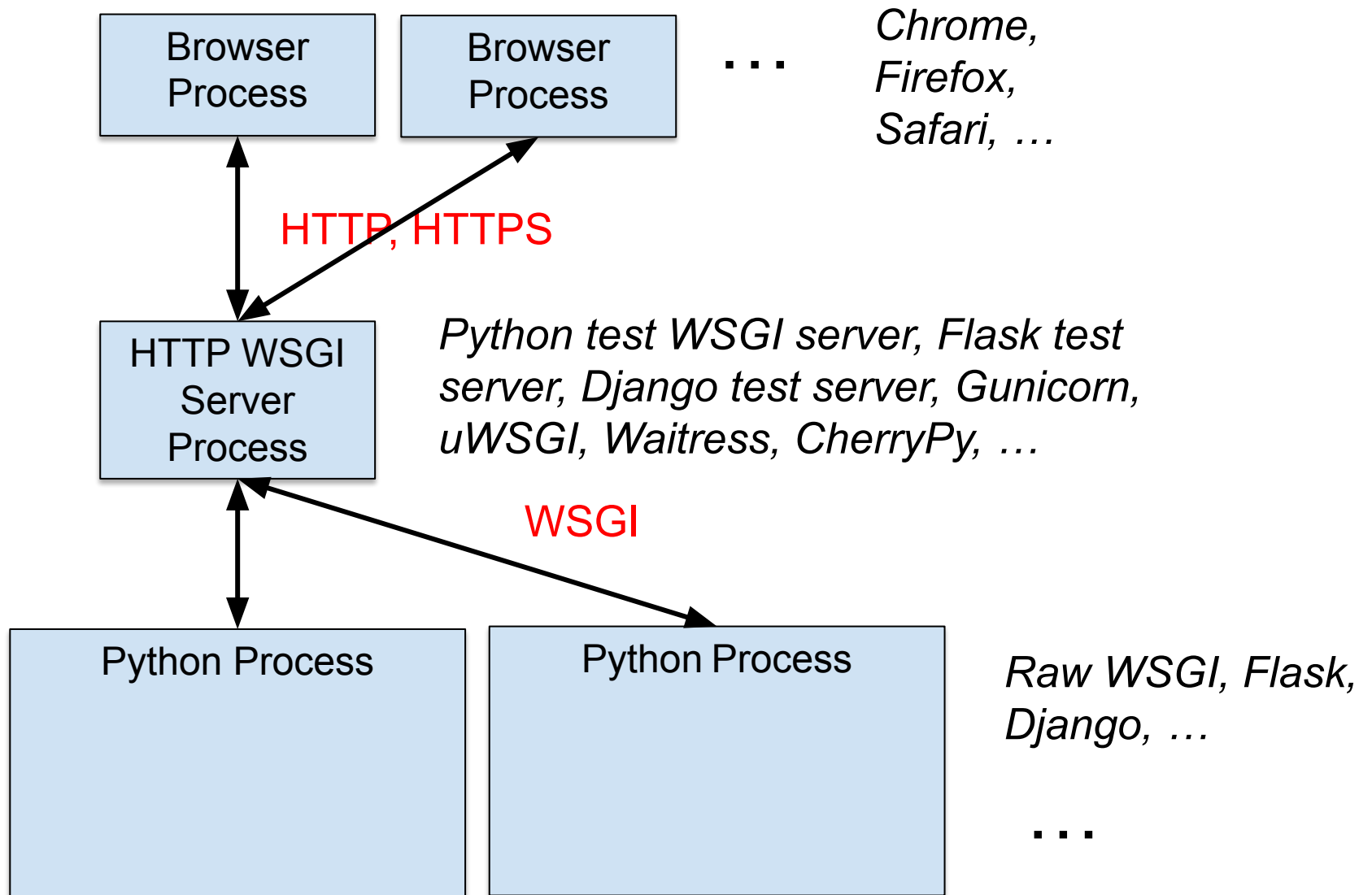
WSGI



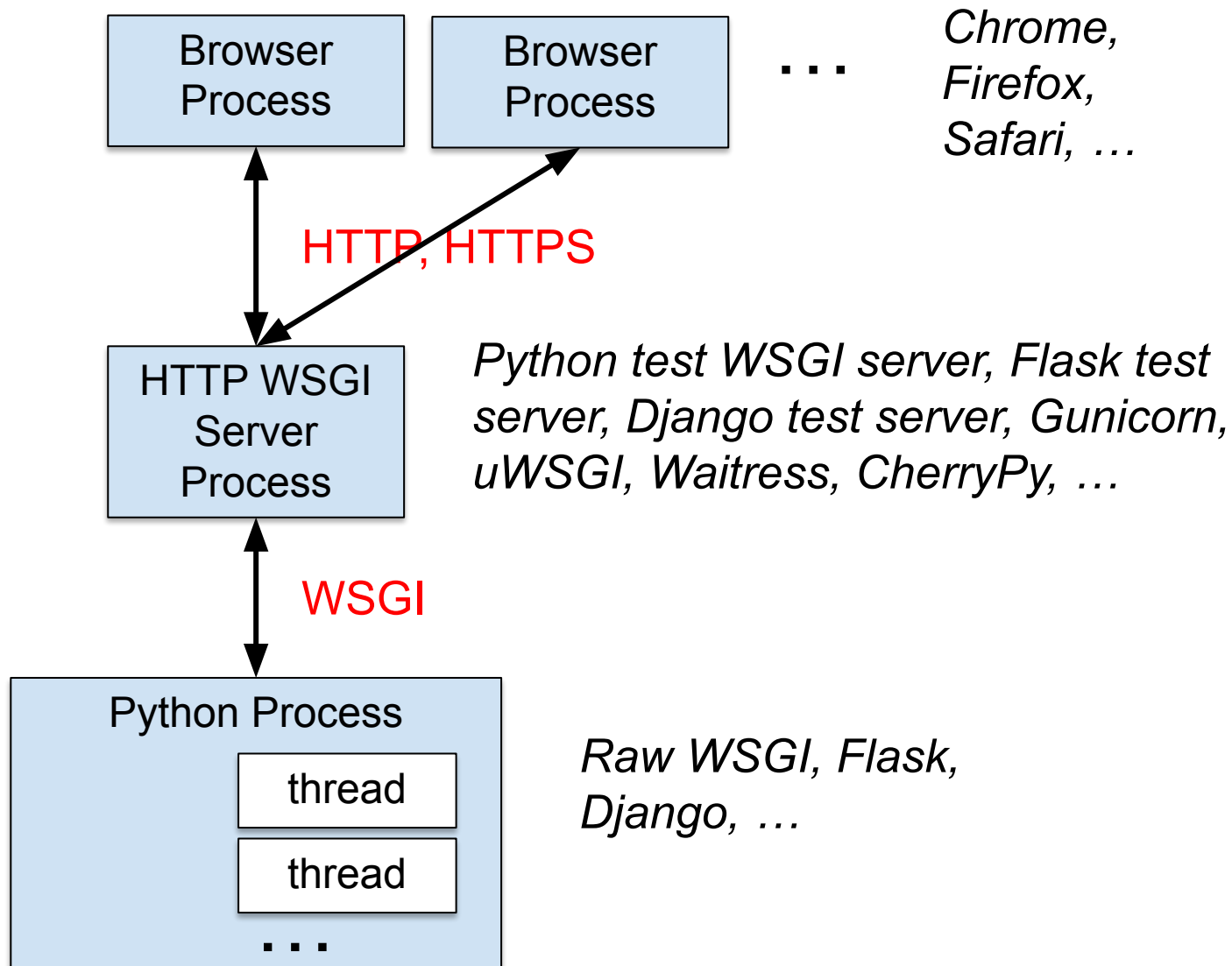
WSGI



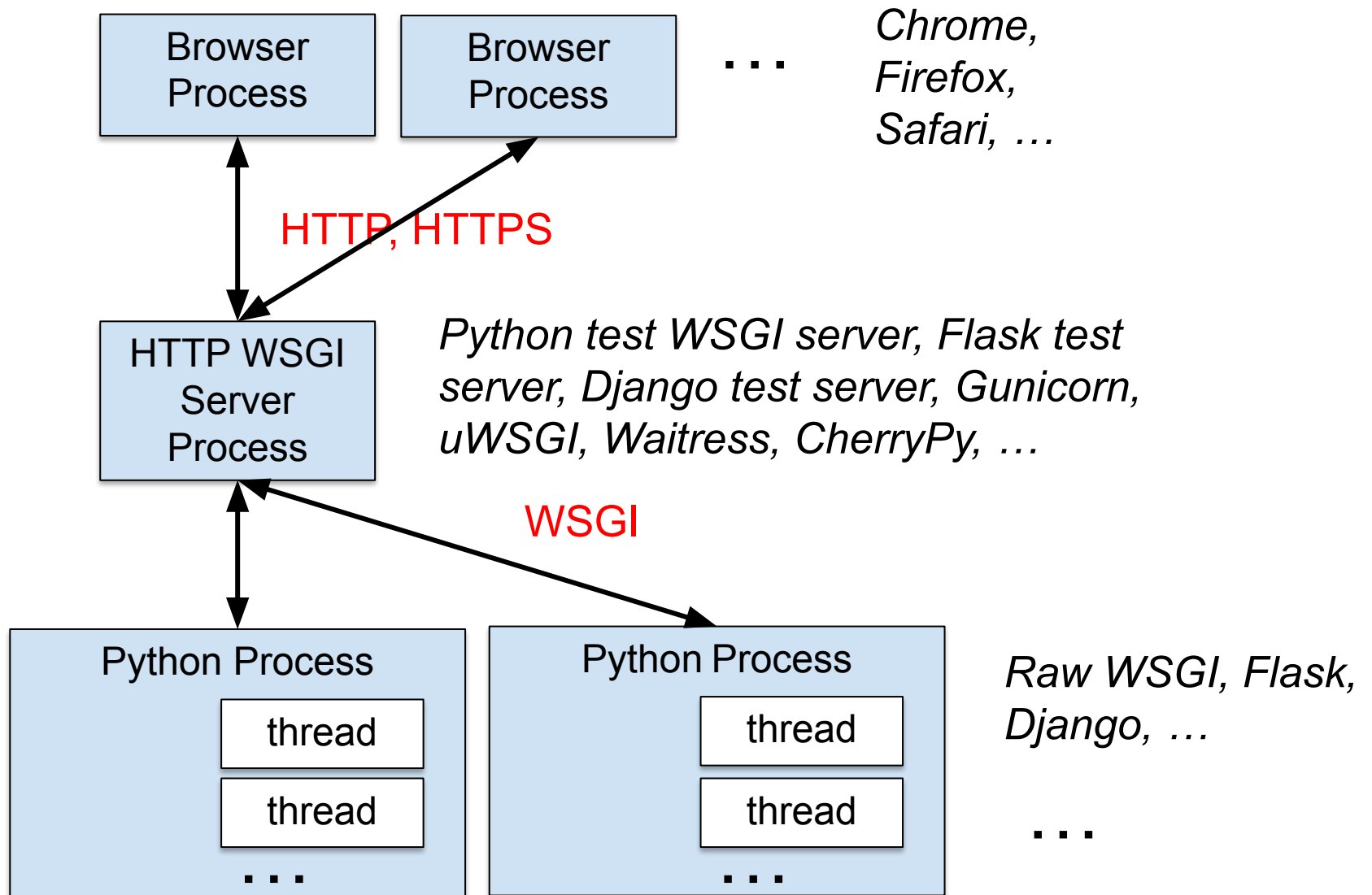
WSGI



WSGI

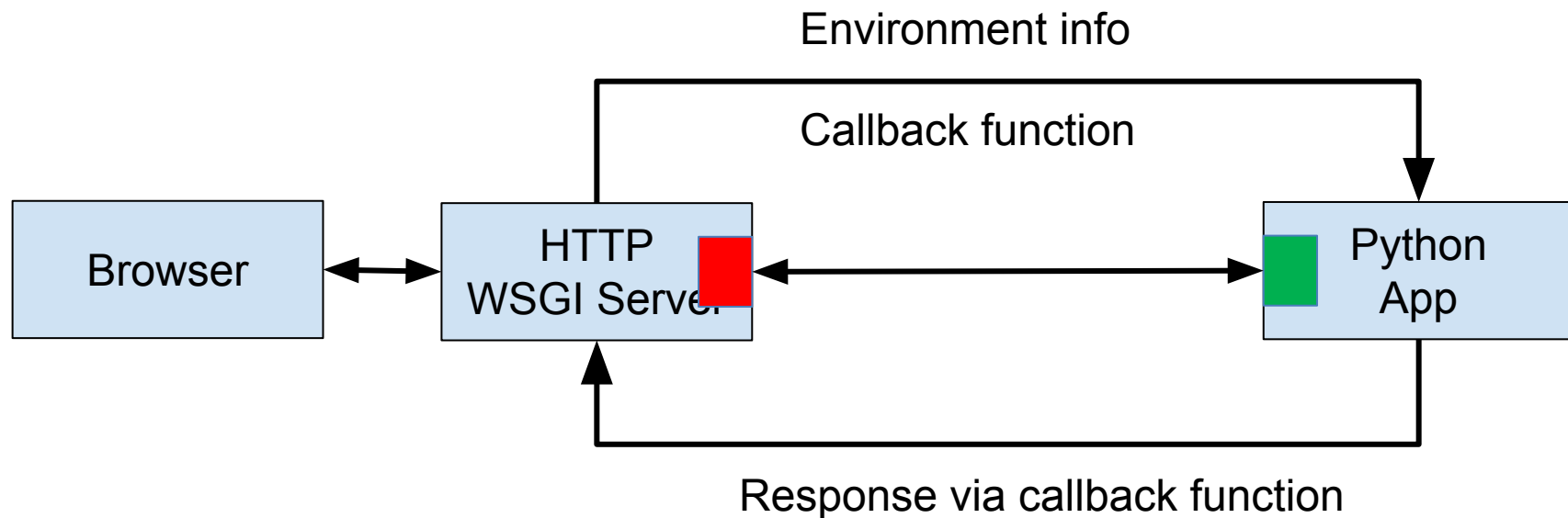


WSGI



WSGI

The Python WSGI Architecture

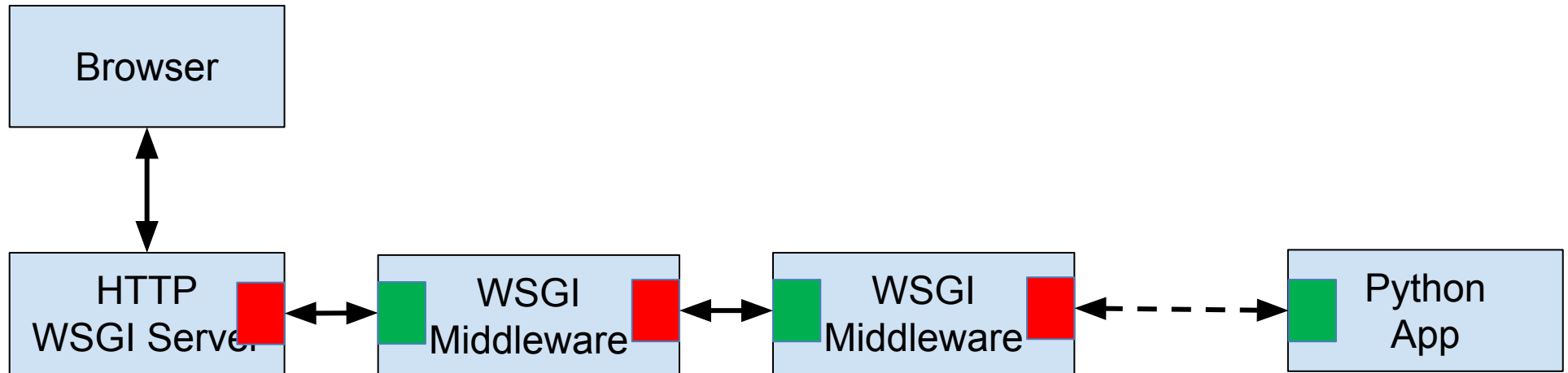


WSGI server-side
Defines messages Python
app can send to HTTP
server

WSGI client-side
Defines messages HTTP server
can send to Python app

Aside: WSGI Middleware

WSGI allows *middleware*



Agenda

- WSGI
- **Using WSGI**
- URL encoding
- HTTP query strings

Using WSGI

- **Question:** How does user command browser to communicate with a HTTP WSGI server?
- **Answer 1:** User enters URL at top of browser

Using WSGI

- Answer 1 elaboration

```
protocol://host:port/someroute
```

The default protocol is http

There is no default host

The default port is 80 for http, 443 for https

A Python program dynamically generates a response specific to *someroute*

Using WSGI

- **Question:** How does user command browser to communicate with a WSGI HTTP server?
- **Answer 1:** User enters URL at top of browser
- **Answer 2:** User clicks on page link

Using WSGI

- Answer 2 elaboration

```
<a href="protocol://host:port/someroute">  
    ...  
</a>
```

The default protocol is the protocol used to deliver this page

The default host is the host that delivered **this** page

The default port is the port from which **this** page was delivered

A Python program dynamically generates a response specific to *someroute*

Using WSGI

- **Question:** How does user command browser to communicate with a WSGI HTTP server?
- **Answer 1:** User enters URL at top of browser
- **Answer 2:** User clicks on page link
- **Answer 3:** User submits a form

Using WSGI

- Answer 3 elaboration

```
<form action="protocol://host:port/someroute" method="get">  
  <input type="submit">  
</form>
```

The default protocol is the protocol used to deliver this page

The default host is the host that delivered **this** page

The default port is the port from which **this** page was delivered

A Python program dynamically generates a response specific to *someroute*

Using WSGI

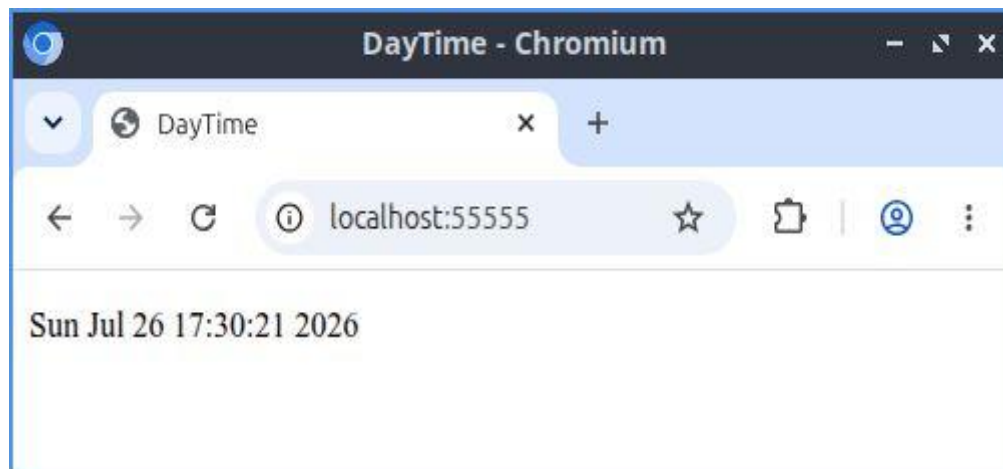
- See **DayTimeWsgi** app

```
$ python runserver.py 55555  
Listening on port 55555
```

Runs the Python
test WSGI server

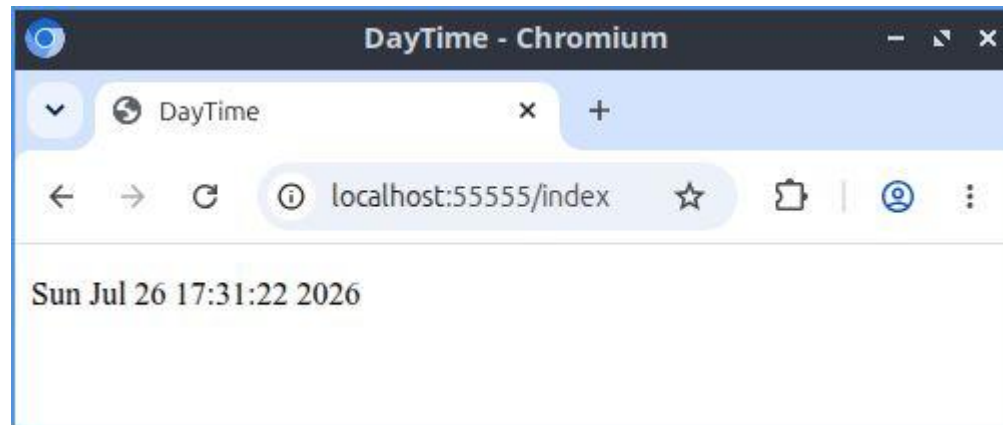
Using WSGI

- See **DayTimeWsgi** app

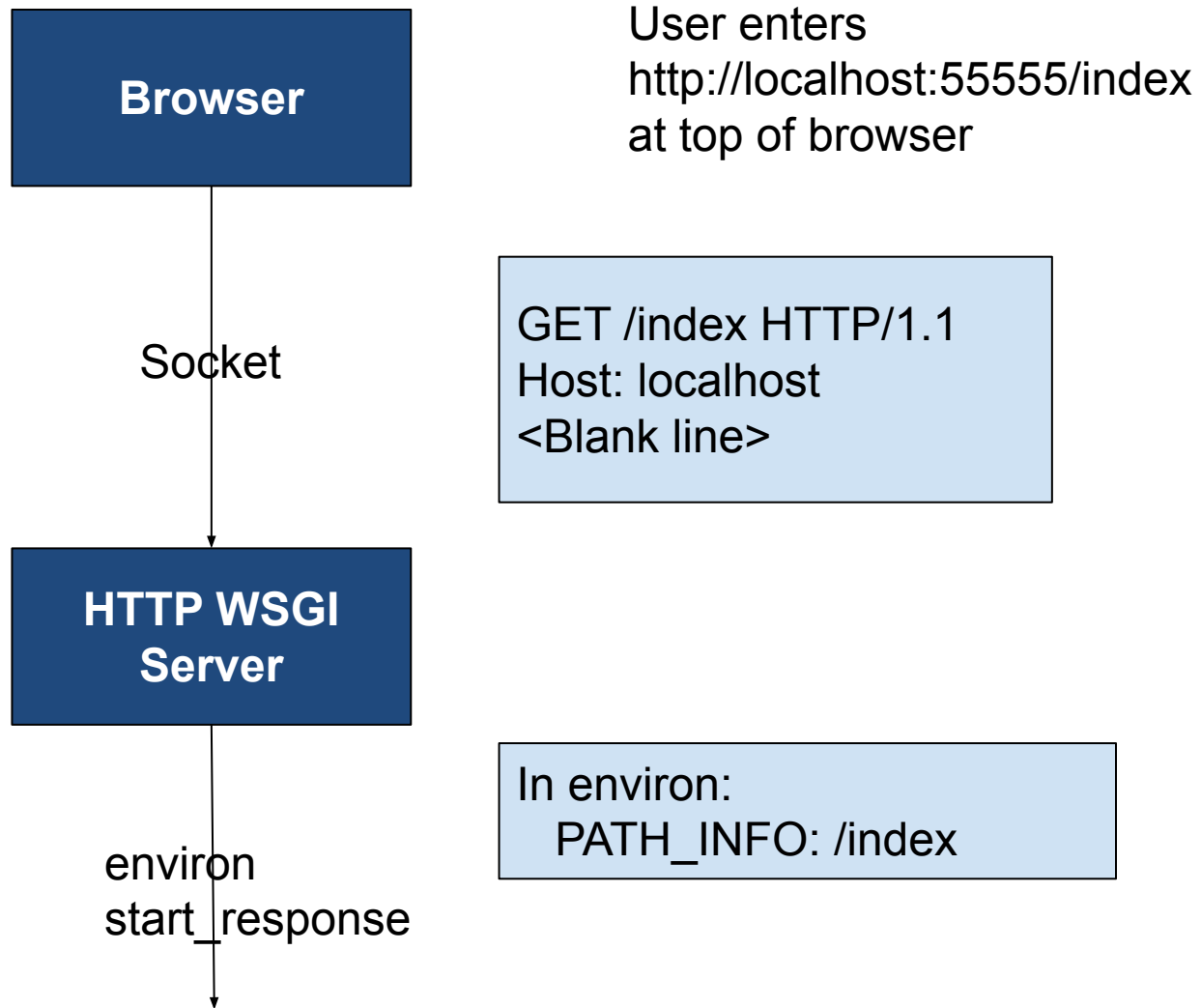


Using WSGI

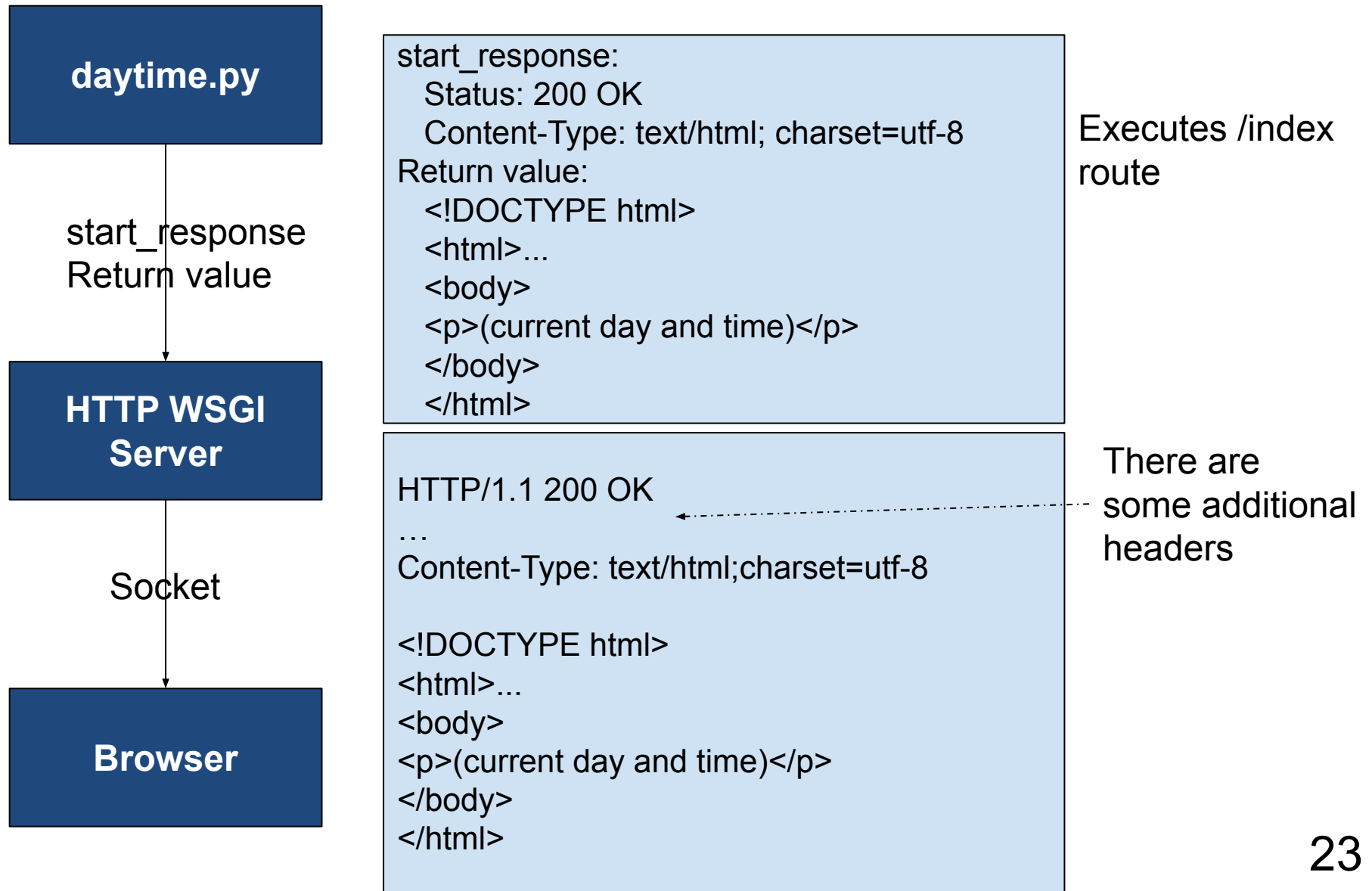
- See **DayTimeWsgi** app
 - Alternative:



Using WSGI



Using WSGI



Using WSGI

- See **DayTimeWsgi** app (cont.)
 - **runserver.py**
 - **daytime.py**

Agenda

- WSGI
- Using WSGI
- **URL encoding**
- HTTP query strings

URL Encoding

- **Problem:**
 - URLs cannot contain spaces or special characters (" , ' , = , & , etc.)
- **Solution:**
 - *URL encoding*

URL Encoding

- ***URL encoding***

- A way of encoding a string such that it may appear within a URL
- Each space is encoded as +
- Each special character is encoded as:
 - %nn where nn are hex digits as per UTF-8, or
 - %nn%nn, or
 - %nn%nn%nn, or
 - %nn%nn%nn%nn

URL Encoding

- Python support for URL encoding
 - `urllib.parse.quote_plus()`
 - string → URL encoded string
 - `urllib.parse.unquote_plus()`
 - URL encoded string → string

URL Encoding

```
$ python
Python 3.14.6 (main, Jun 11 2026,
12:32:48) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> import urllib.parse
>>> urllib.parse.quote_plus('one')
'one'
>>> urllib.parse.quote_plus('t wo')
't+wo'
>>> urllib.parse.quote_plus('th"r ee')
'th%22r+ee'
>>> quit()
$
```

Agenda

- WSGI
- Using WSGI
- URL encoding
- **HTTP query strings**

HTTP Query Strings

- **Problem:**

- Often we want:

- Client/browser provides arguments to HTTP WSGI server
 - HTTP WSGI server passes arguments to Python app
 - Python app dynamically generates response based upon those arguments

- **Solution:**

- *Query strings*

- Key=value pairs appended to URL

HTTP Query Strings

- **Question:** How does browser pass arguments to HTTP server?
- **Answer 1:** User enters URL at top of browser with a query string appended

HTTP Query Strings

- Answer 1 elaboration

```
protocol://host:port/somroute?key1=value1&key2=value2
```

HTTP Query Strings

- **Question:** How does browser pass arguments to HTTP server?
- **Answer 1:** User enters URL at top of browser with a **query string** appended
- **Answer 2:** User clicks on page link whose `href` attribute has a **query string** appended

HTTP Query Strings

- Answer 2 elaboration

```
<a href="protocol://host:port/someroute?  
    key1=value1&key2=value2">  
    ...  
</a>
```

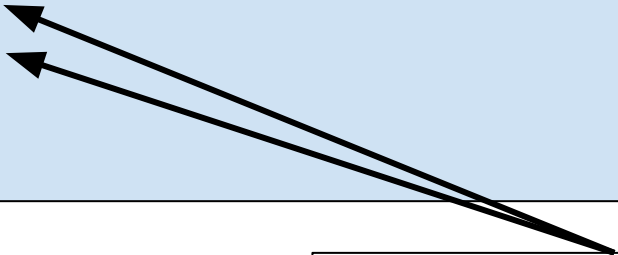
HTTP Query Strings

- **Question:** How does browser pass arguments to HTTP server?
- **Answer 1:** User enters URL at top of browser with a query string appended
- **Answer 2:** User clicks on page link whose `href` attribute has a **query string** appended
- **Answer 3:** User submits form that has `input` elements of type `text`

HTTP Query Strings

- Answer 3 elaboration

```
<form action="protocol://host:port/somerooute" method="get">  
  <input type="text" name="key1">  
  <input type="text" name="key2">  
  <input type="submit">  
</form>
```

Two black arrows originate from a text box on the right and point to the two text input elements in the HTML code. The first arrow points to the `<input type="text" name="key1">` element, and the second arrow points to the `<input type="text" name="key2">` element.

User enters *value1* and *value2* in input elements

Sends HTTP request as if generated by clicking on page link with href
`protocol://host:port/somerooute?key1=value1&key2=value2`

HTTP Query Strings

- Answer 3 elaboration (cont.)

```
<form action="protocol://host:port/somerooute" method="get">  
  <input type="text" name="key1" value="initvalue1">  
  <input type="text" name="key2" value="initvalue2">  
  <input type="submit" value="somelabel">  
</form>
```

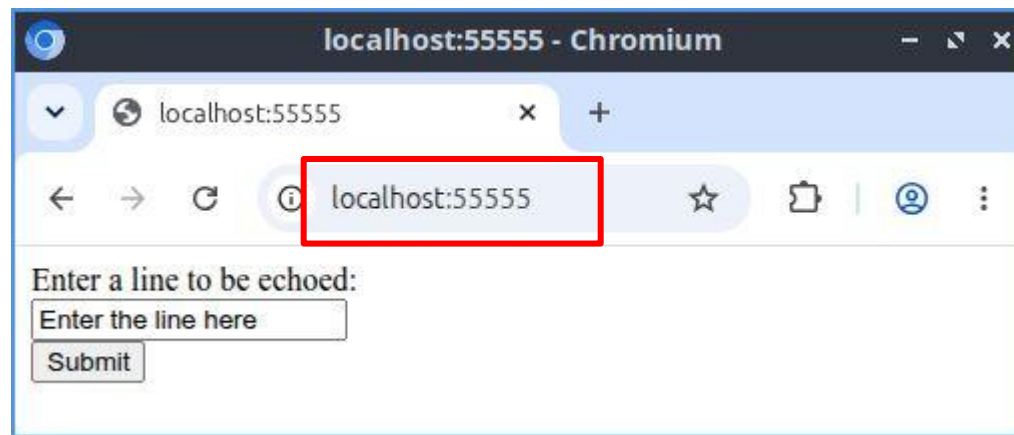
HTTP Query Strings

- See **EchoWsgiGet** app

```
$ python runserver.py 55555  
Listening on port 55555
```

HTTP Query Strings

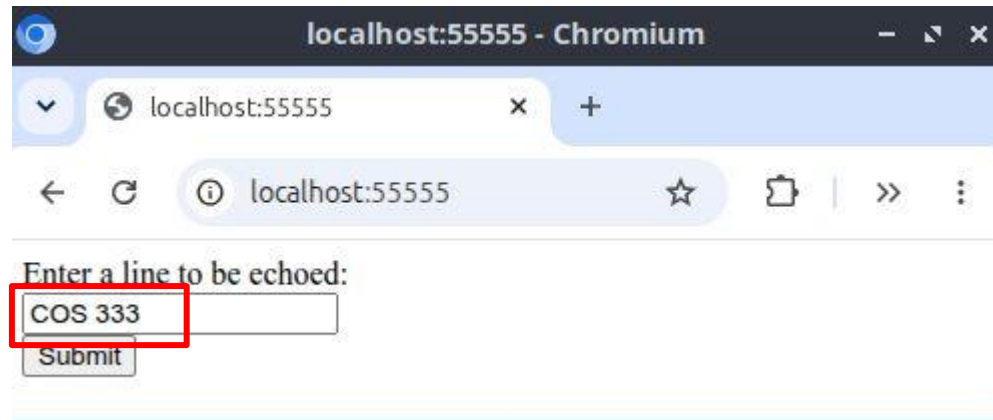
- See EchoWsgiGet app (cont.)



index page

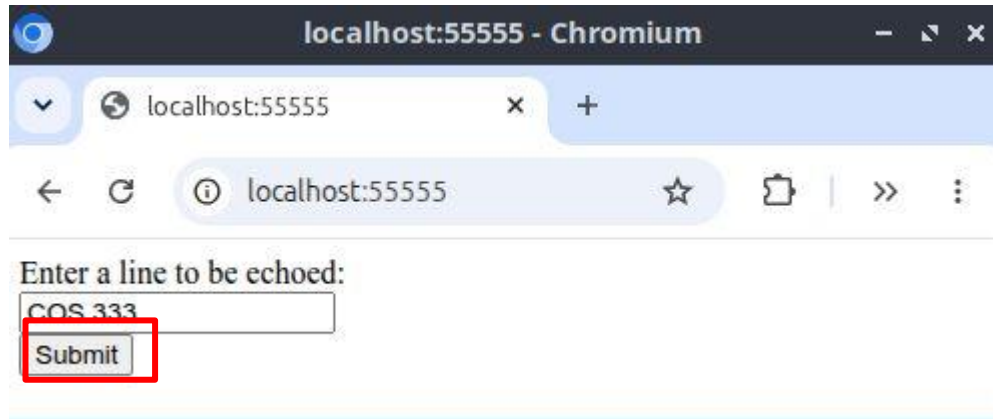
HTTP Query Strings

- See EchoWsgiGet app (cont.)



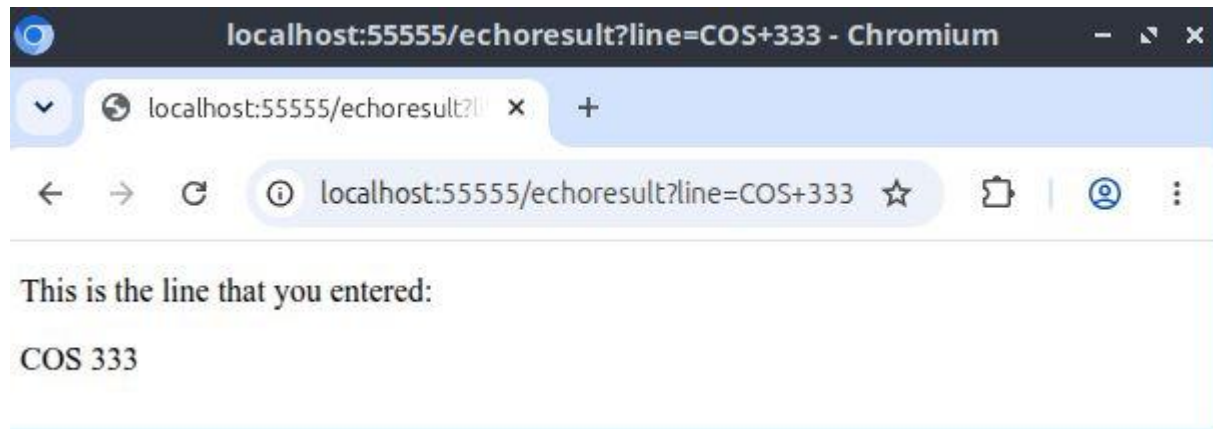
HTTP Query Strings

- See EchoWsgiGet app (cont.)



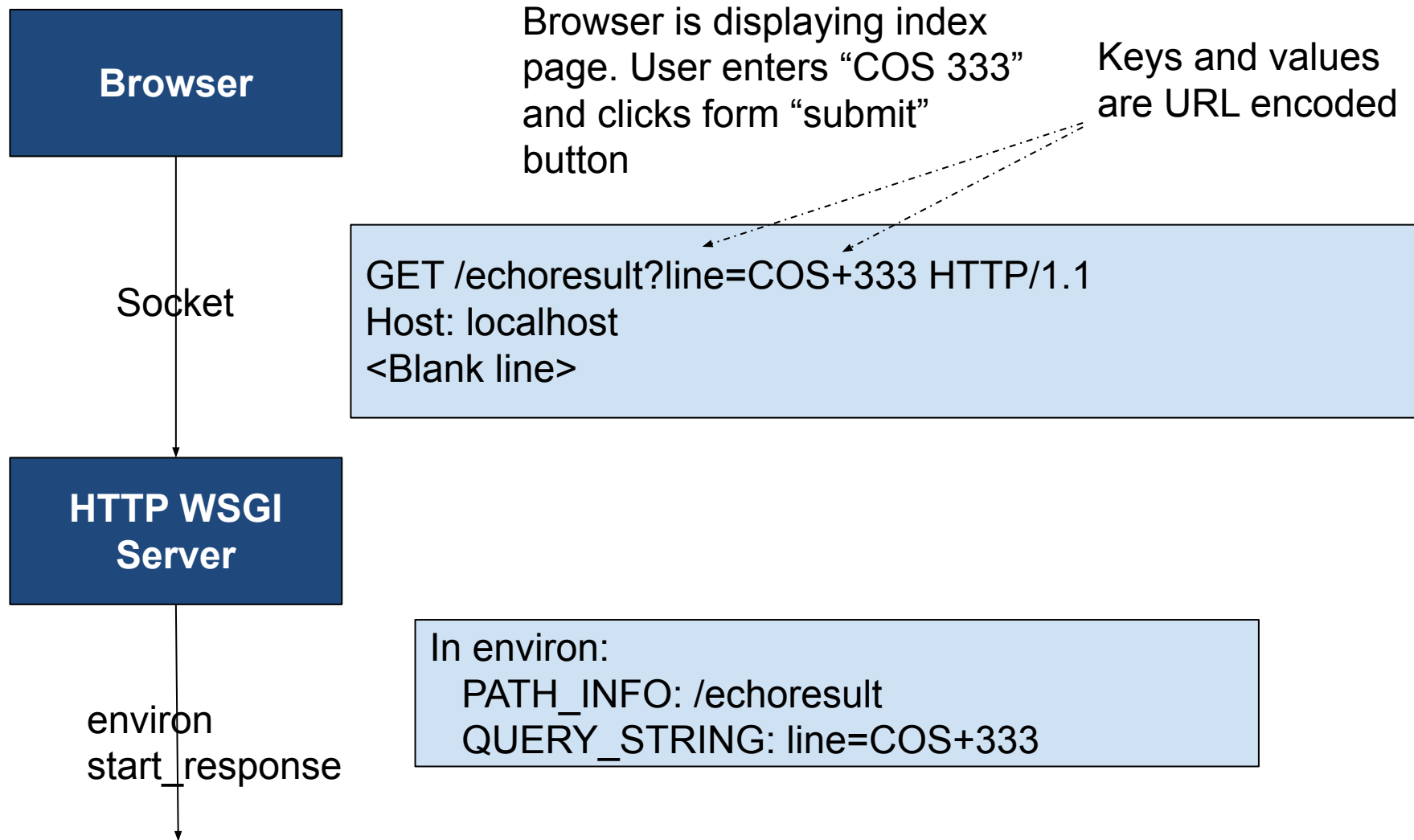
HTTP Query Strings

- See **EchoWsgiGet** app (cont.)

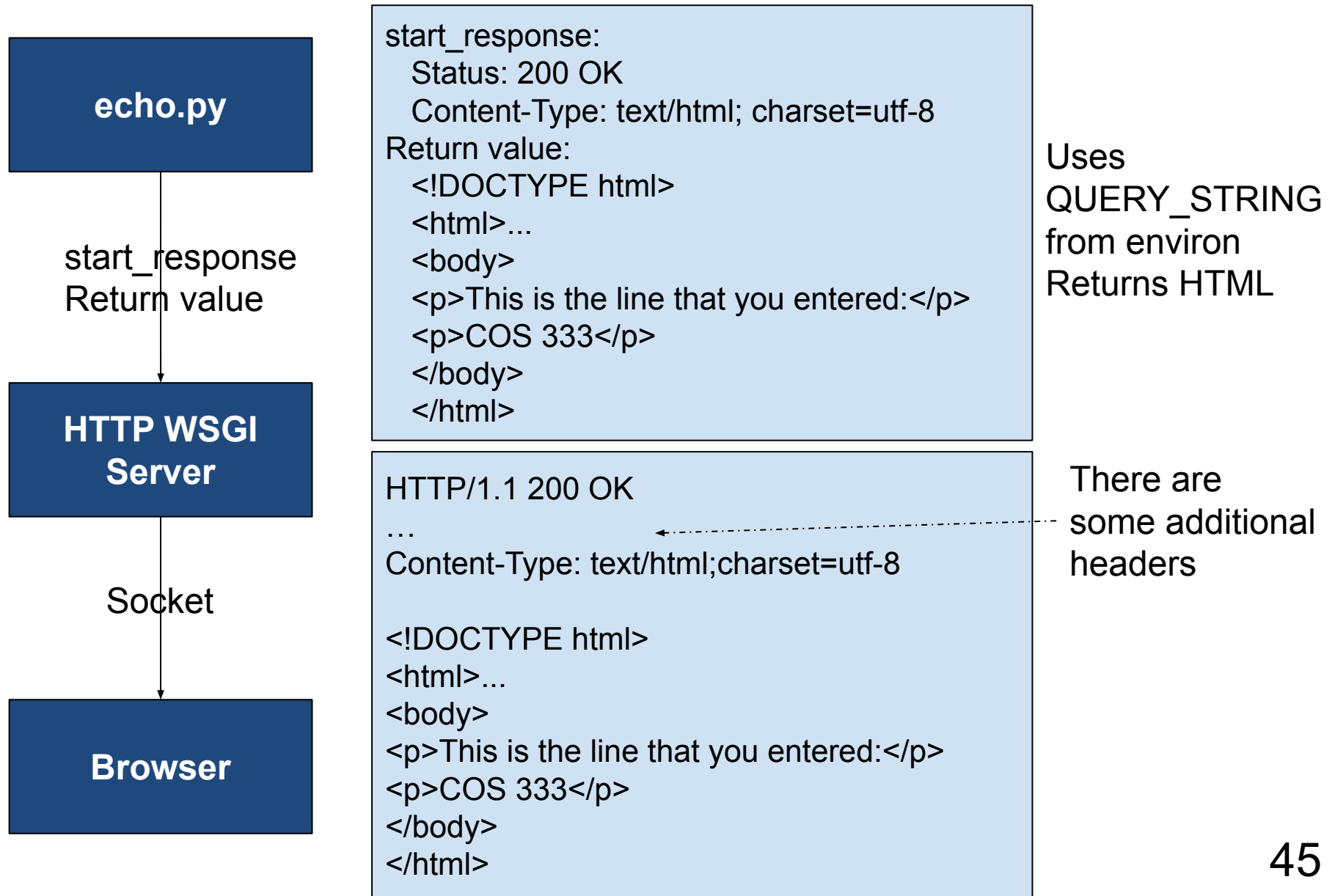


echoresult page

HTTP Query Strings



HTTP Query Strings



HTTP Query Strings

- See **EchoWsgiGet** app
 - **runserver.py**
 - **echo.py**

Summary

- We have covered:
 - Python WSGI programming
 - Sending arguments to WSGI programs